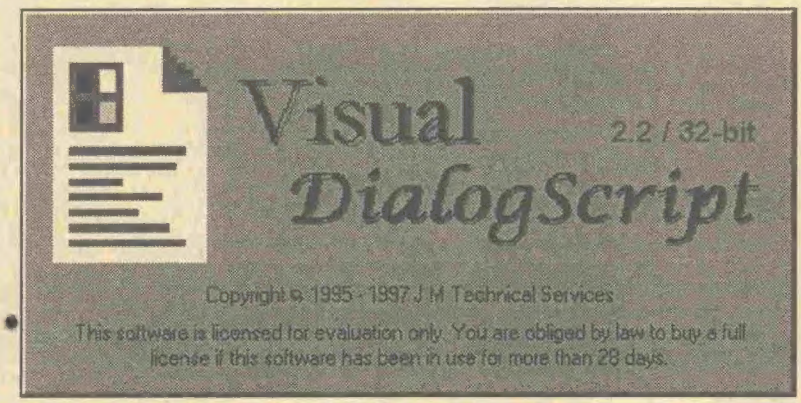
ธาวินทร์ ลีลาคุณารักษ์

SOFT MAIL

Visual DialogScript 2.2 เครื่องมือสร้างโปรแกรมอย่างง่าย ๆ



วิธีการสร้างชุดคำสั่งของ Visual DialogScript สำหรับสร้างโปรแกรมเก็บชื่อและเบอร์โทรศัพท์

ถ้าหากท่านผู้อ่านเคยเขียนโปรแกรมที่ทำงานบนดอสมาบ้างแล้ว คงจะรู้จักการทำแบตช์ไฟล์ (Batch File) เป็นอย่างดี ซึ่งแบตช์ไฟล์นั้นเป็นเสมือนโปรแกรมคอมพิวเตอร์ขนาดเล็ก ๆ สำหรับเรียกโปรแกรมหรือจัดสภาพแวดล้อมให้กับโปรแกรมที่จะเรียกขึ้นมาทำงาน แต่เมื่อเราใช้ระบบปฏิบัติการวินโดวส์ 95 แทน เราจะไม่ค่อยได้เกี่ยวข้องกับแบตช์ไฟล์มากนัก เนื่องจากวินโดวส์ 95 พยายามจะจัดองค์ประกอบของระบบด้วยตนเอง และโปรแกรมต่าง ๆ ที่ทำงานบนวินโดวส์ 95 ส่วนใหญ่จะมีการจัดสภาพแวดล้อมให้เหมาะสมกับการทำงานอยู่แล้ว

Visual DialogScript 2.2

เนื่องจากคุณลักษณะของการใช้คำสั่งที่เรียบง่ายของแบตช์ไฟล์ ทำให้มีคนสร้างคอมไพเลอร์สำหรับแปลงต้นฉบับโปรแกรมที่ใช้ชุดคำสั่งง่าย ๆ เหมือนกับการเขียนแบตช์ไฟล์ขึ้นมา ซึ่งโปรแกรม Visual DialogScript ก็เป็นคอมไพเลอร์ตัวหนึ่งที่ใช้หลักการดังกล่าวนี้

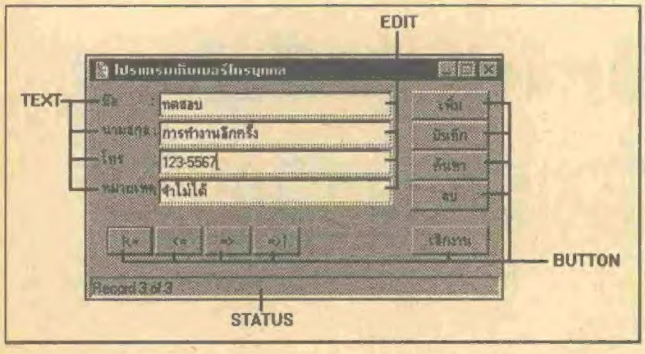
วิธีใช้ Visual DialogScript สร้างโปรแกรมคอมพิวเตอร์จะคล้ายกับคอมไพเลอร์ที่ใช้หลักการ Visual Programming ทั่วไป แต่ Visual DialogScript นั้นไม่ถือว่าเป็นคอมไพเลอร์แบบ Visual Programming โดยตรง เนื่องจากเราจะใช้เครื่องมือสร้างหน้าจอที่เรียกว่า Dialog Editor สร้างองค์ประกอบส่วนหน้าจอของโปรแกรมที่เราจัดทำได้เฉพาะตอนลงมือสร้างครั้งแรกเท่านั้น หลังจากที่ Visual DialogScript แปลงงานเป็นต้นฉบับโปรแกรม (Source Code) แล้ว เราจะไม่สามารถกลับไปทำงานใน Visual Mode ได้อีก

ภาษาโปรแกรมที่ใช้

แต่อย่างไรก็ดี ชุดคำสั่งของภาษาโปรแกรมที่ใช้กับ Visual DialogScript นั้น จะเป็นชุดคำสั่งแบบพื้น ๆ ง่าย ๆ ผู้ที่ใช้ Visual DialogScript สร้างโปรแกรมไม่จำเป็นต้องเป็นโปรแกรมเมอร์ ก็สามารถสร้างโปรแกรมต่าง ๆ ออกมาได้อย่างง่ายดาย

ลักษณะทั่วไปของภาษาโปรแกรมที่ใช้กับ Visual DialogScript เป็นภาษาโปรแกรมที่ไม่มีโครงสร้าง

รูปที่ 1 หน้าตาของโปรแกรม Visual DialogScript



รูปที่ 2 โปรแกรมที่ทดลองสร้างด้วย Visual DialogScript

(Non-Struture Programming) โดยจะมีองค์ประกอบพื้นฐานอยู่ 4 อย่าง คือ %Variable,:Label,Command และ @Function ซึ่งมีรายละเอียดคร่าว ๆ ดังนี้

- **Variable** หมายถึง การกำหนดตัวแปร ซึ่งวิธีการสร้างตัวแปรใน Visual DialogScript จะต้องขึ้นต้นด้วยเครื่องหมาย "%" แล้วตามด้วย 0-9 หรือ A-Z ตัวแปรที่เรากำหนดขึ้นนี้ จะสามารถเก็บข้อมูลได้ทุกประเภท ในโปรแกรมชุดหนึ่ง ๆ จะมีตัวแปรได้ไม่เกิน 35 ตัว

- **:Label** หมายถึง การระบุจุดที่จะให้โปรแกรมกระโดดไปกระทำคำสั่ง ด้วยคำสั่ง GOTO หรือ GOSUB โดยการสร้างลาเบลจะต้องนำหน้าด้วย ":" แล้วตามด้วยชื่อที่จะใช้อ้างอิง ลาเบลจะมี 2 ลักษณะ คือ ลาเบลแบบทั่วไป จะปิดท้ายช่วงคำสั่งด้วยคำสั่ง GOTO และลาเบลแบบ SUB PROGRAM จะปิดท้ายช่วงของคำสั่งด้วยคำสั่ง EXIT

- **Command** หมายถึง คำสั่งทั่วไปที่ใช้ใน Visual DialogScript การเขียน Command ไม่ต้องมีตัวอักษรพิเศษนำหน้าเหมือนองค์ประกอบอื่น ๆ ของภาษา

- **@Function** หมายถึง ชุดคำสั่งที่ให้ค่าใดค่าหนึ่งออกมา ในการเขียนฟังก์ชันจะต้องขึ้นต้นด้วยเครื่องหมาย "@" แล้วตามด้วยชื่อของฟังก์ชันที่เราเรียกใช้

การประยุกต์ใช้งาน

เนื่องจากภาษาโปรแกรมที่ใช้กับ Visual DialogScript จะมีลักษณะเหมือนการสร้างแบตช์ไฟล์ หรือคล้ายกับการเขียนคำในภาษาอังกฤษทั่วไป จะไม่มีความยืดหยุ่นเหมือนกับภาษาโปรแกรมอย่าง C, Pascal ทำให้ Visual DialogScript ไม่เหมาะที่ใช้สร้างโปรแกรมคอมพิวเตอร์ขนาดใหญ่ แต่ Visual DialogScript จะเหมาะสำหรับการสร้างโปรแกรมขนาดเล็ก เช่นใช้สร้างโปรแกรมติดตั้งโปรแกรมประยุกต์ หรือใช้สร้างโปรแกรมเพื่อแสดงกรอบโต้ตอบ (Dialog) อย่างง่าย ๆ หรือใช้สร้างกรอบเมนูเพื่อเรียกโปรแกรมอื่น ๆ ขึ้นมาทำงานเป็นต้น

ทดลองสร้างโปรแกรม

เพื่อให้เกิดความคุ้นเคยกับการใช้ Visual DialogScript สร้างโปรแกรมคอมพิวเตอร์ ผมมีตัวอย่างโปรแกรมง่าย ๆ ให้ทดลองสร้างอยู่ชุดหนึ่ง ซึ่งเป็นโปรแกรมที่ใช้สำหรับเก็บชื่อและเบอร์โทรศัพท์ของบุคคลที่เรารู้จัก หน้าตาของโปรแกรมที่จะจัดทำอยู่ในรูปที่ 2

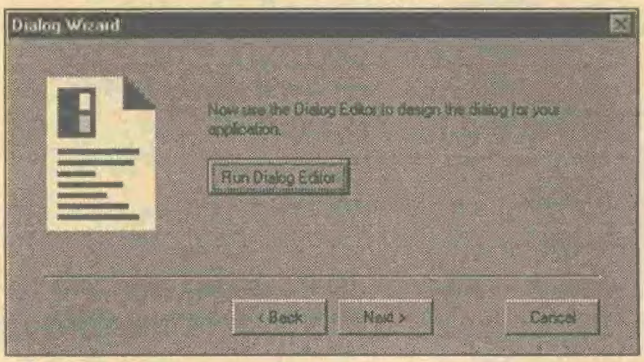
ในการลงมือสร้างโปรแกรมนั้น เราจะแบ่งงานออกเป็น 2 ส่วน คือ การสร้างส่วนหน้าจอของโปรแกรม และการสร้างชุดคำสั่งเพิ่มเติมลงในต้นฉบับของโปรแกรม โดยจะมีรายละเอียดดังนี้

การสร้างส่วนหน้าจอของโปรแกรม

การสร้างหน้าจอของโปรแกรมที่สร้างด้วย Visual DialogScript จะทำได้ 2 วิธี คือ ใช้ Dialog Editor จัดองค์ประกอบของหน้าจอใน Visual Mode ก่อน แล้วค่อยแปลงเป็นต้นฉบับโปรแกรมในภายหลัง หรือจะสร้างชุดคำสั่งขึ้นมาเองทั้งหมด โดยไม่ต้องใช้ Dialog Editor ก็สามารถทำได้เช่นเดียวกัน

สำหรับการสร้างหน้าจอของโปรแกรมโดยใช้ Dialog Editor จะเริ่มจากเลือกที่หัวข้อ "File" จากเมนูหลักของโปรแกรม แล้วเลือกหัวข้อ "New" และ "Dialog Wizard" ตามลำดับ จากนั้นเราจะดำเนินการตามลำดับขั้นตอนที่ Dialog Wizard กำหนดไว้จนถึงขั้นตอนที่ปรากฏปุ่มคำสั่ง "Run Dialog Editor" อยู่บน Wizard (ตัวอย่างในรูปที่ 3) เมื่อใช้เม้าส์คลิกที่ปุ่มคำสั่งดังกล่าว Visual DialogScript จะเรียก Dialog Editor ขึ้นมาทำงานพร้อมกับมีเฟรมสำหรับสร้างหน้าจอปรากฏขึ้นมาด้วย

บนกรอบหน้าต่าง Dialog Editor จะประกอบด้วยลิ้นชีตจำนวน 2 ชุด คือ ลิ้นชีต "Dialog" ซึ่งเป็นที่เก็บรายละเอียดของหน้าจอที่เราจัดทำทั้งหมด และลิ้นชีต "Dialog Element" จะเป็นส่วนที่ใช้กำหนดรายละเอียดของชิ้นส่วนแต่ละชิ้นส่วนที่เรานำมาประกอบเป็นหน้าจอของโปรแกรม ลิ้นชีตชุดนี้จะปรากฏขึ้นอยู่ด้านหน้าทันทีที่เราคลิกที่รูปของชิ้นส่วนที่เราสนใจ บนเฟรมสำหรับสร้างหน้าจอของโปรแกรม

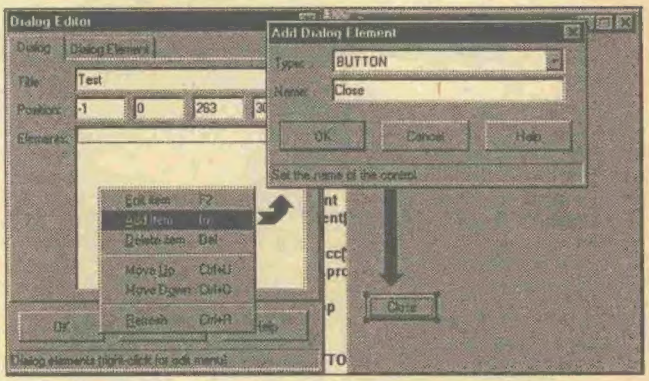


รูปที่ 3 Dialog Wizard

ในรูปที่ 4 เป็นขั้นตอนการสร้างชิ้นส่วนของโปรแกรม เพื่อใช้เป็นส่วนประกอบของโปรแกรมที่เราจัดทำ โดยขั้นตอนจะเริ่มจากใช้เม้าส์คลิกที่ช่อง Element ของลิ้นชัก "Dialog" กดปุ่มขวาของเม้าส์ แล้วเลือกหัวข้อ Add Item จากเมนูที่ปรากฏขึ้นตามลำดับ โปรแกรม Visual DialogScript จะแสดงไดอะล็อก "Add Dialog Element" ขึ้นมาให้เราเลือกชนิด (Type) และกำหนดชื่อ (Name) ของชิ้นส่วนที่เราจะสร้างขึ้น และเมื่อเราคลิกที่ปุ่มคำสั่ง OK ของไดอะล็อกดังกล่าว บนเฟรมสำหรับสร้างหน้าจอจะมีชิ้นส่วนโปรแกรมที่เรากำหนดไว้ปรากฏขึ้นมาทันที

สำหรับส่วนประกอบแบบต่าง ๆ เราสามารถนำมาใส่บนหน้าจอของโปรแกรมที่สร้างด้วย Visual DialogScript ได้ จะมีรายละเอียดดังตารางที่ 1 ส่วนในตารางที่ 2 จะเป็นชิ้นส่วนที่เราจะใช้สร้างโปรแกรม ซึ่งจะมีรายละเอียดการตั้งชื่อและแก้ไขคุณสมบัติของชิ้นส่วนด้วย

หลังจากที่เราจัดองค์ประกอบของโปรแกรมครบตามที่เราต้องการแล้ว เมื่อใช้เม้าส์คลิกที่ปุ่มคำสั่ง OK ของ Dialog Editor และคลิกที่ปุ่มคำสั่ง "Finite" ของ Dialog Wizard ตามลำดับ โปรแกรม Visual DialogScript จะทำการแปลงงานทั้งหมดเป็นชุดคำสั่งของ Visual DialogScript ให้ทันที สำหรับโปรแกรมที่เราจัดทำนั้น ในเบื้องต้นนี้จะได้ต้นฉบับโปรแกรมออกมาดังรายละเอียดในลิสติงที่ 1 ซึ่งเราจะต้องนำมาสร้างชุดคำสั่งเพิ่มเติม เพื่อให้โปรแกรมทำงานตามที่เรากำหนดต่อไป



รูปที่ 4 ขั้นตอนการสร้างองค์ประกอบของโปรแกรมด้วย Dialog Editor

ตารางที่ 1 ส่วนประกอบที่ใช้สร้างหน้าจอของ Visual DialogScript

ส่วนประกอบ			ใช้สำหรับ	
BITMAP	BUTTON		สร้างพื้นที่วางรูปภาพ	สร้างปุ่มคำสั่ง
CHECK	COMBO		สร้าง Check Box	สร้าง Combo Box
DLGTYPE	EDIT	LIST	กำหนดชนิดของหน้าจอ	สร้างช่องรับข้อมูล
POPUP	PROGRESS		สร้าง List Box	กำหนด Popup Menu
RADIO	STATUS	STYLE	สร้างแถบวัดปริมาณ	สร้าง Radio Button
TASKION	TEXT		สร้างแถบสถานะ กำหนดรูปแบบชิ้นส่วนต่าง ๆ	
			กำหนด Icon ไว้ที่ Task Bar (Windows95) สร้างพื้นที่แสดงข้อความ	

ตารางที่ 2 ชิ้นส่วนที่ใช้สร้างโปรแกรมตัวอย่าง

ชิ้นส่วน		คุณสมบัติ	ค่าที่กำหนดใหม่		
TEXT [ชื่อ]	EDIT [ชื่อ]	Caption	"ชื่อ :"	NAME	
TEXT [นามสกุล]	EDIT [นามสกุล]	Name	"นามสกุล :"	LAST	NAME
TEXT [โทร] EDIT	[โทร] TEXT	Caption	"โทร :"	TEL	
[หมายเหตุ] EDIT	[หมายเหตุ]	Name	"หมายเหตุ"	NOTE	Home
BUTTON [|<=]		Caption	"|<="	Prev	
BUTTON [<]		Name	"<"	Next	">"
BUTTON [>]		Caption	End	"=>|"	Add
BUTTON [=>|]		Name	"เพิ่ม"	Post	"บันทึก"
BUTTON [เพิ่ม]		Caption	Del	"ลบ"	Quit
BUTTON [บันทึก]		Name	"เลิกงาน"	SP	
BUTTON [ลบ]		Caption			
BUTTON [เลิกงาน]		Name			
STATUS		Caption			
		Name			
		Caption			
		Name			
		Caption			
		Name			
		Caption			
		Name			
		Caption			
		Name			
		Caption			
		Name			

การสร้างชุดคำสั่งเพิ่มเติม

ในลิสติงที่ 2 เป็นต้นฉบับโปรแกรม (Source Code) ที่สมบูรณ์ของโปรแกรมตัวอย่าง ซึ่งจะสังเกตเห็นว่าชุดคำสั่งที่เราเติมลงไปนั้น ส่วนใหญ่จะเป็นงานที่เกี่ยวข้องกับปุ่มคำสั่งที่เราเตรียมไว้ก่อนหน้านี้แทบทั้งสิ้น โดยงานที่ปุ่มคำสั่งเหล่านี้ Visual DialogScript จะเตรียมลาเบลพร้อมกับคำแนะนำเบื้องต้นไว้ให้ จะมีเพียงบรรทัดที่ 023 ถึง 033 และช่วง :updaterec (บรรทัดที่ 145 ถึง 159) กับช่วง :showrec (บรรทัดที่ 161 ถึง 169) เท่านั้น ที่เราสร้างขึ้นมาเองทั้งหมด สำหรับรายละเอียดของชุดคำสั่งในช่วงต่าง ๆ จะมีรายละเอียดดังนี้

- ชุดคำสั่งในช่วง บรรทัดที่ 023 ถึง 033 ชุดคำสั่งที่เราสร้างขึ้นในช่วงนี้ จะเป็นชุดคำสั่งที่ทำงานหลังวินโดวส์วาดส่วนหน้าจอของโปรแกรมที่เราจัดทำเสร็จ โดยจะทำหน้าที่เตรียมพื้นที่เก็บข้อมูลในหน่วยความจำแบบเรียงข้อมูลอัตโนมัติ (คำสั่ง List

ลิสติงที่ 1 untitle.dsc

```
001: title Test
002: DIALOG CREATE,โปรแกรมเก็บเบอร์โทรบุคคล,-1,0,344,185,-
003: TEXT(TEXT1;8;10;;;ชื่อ:),-
004: EDIT(NAME;8;56;192),-
005: TEXT(TEXT2;32;10;;;นามสกุล :),-
006: EDIT(LASTNAME;32;56;192),-
007: TEXT(TEXT3;56;10;;;โทร :),-
008: EDIT(TEL;56;56;192),-
009: TEXT(TEXT4;80;10;;;หมายเหตุ),-
010: EDIT(NOTE;80;56;192),-
011: BUTTON(Home;120;14;38;;|<=),-
012: BUTTON(Prev;120;54;38;;<=),-
013: BUTTON(Next;120;94;38;;=>),-
014: BUTTON(End;120;134;38;;=>|),-
015: BUTTON(Add;8;264;;;เพิ่ม),-
016: BUTTON(Post;33;264;;;บันทึก),-
017: BUTTON(Find;58;264;;;ค้นหา),-
018: BUTTON(Del;83;264;;;ลบ),-
019: BUTTON(Quit;120;264;;;เลิกงาน),-
020: Status(SP;No data)
021: :evloop
022: wait event
023: goto @event()
024: :HomeBUTTON
025: info Replace this line with code to process the HomeBUTTON event
026: goto evloop
027: :PrevBUTTON
028: info Replace this line with code to process the HomeBUTTON event
029: goto evloop
030: :NextBUTTON
031: info Replace this line with code to process the HomeBUTTON event
032: goto evloop
033: :EndBUTTON
034: info Replace this line with code to process the HomeBUTTON event
035: goto evloop
036: :AddBUTTON
037: info Replace this line with code to process the HomeBUTTON event
038: goto evloop
039: :PostBUTTON
040: info Replace this line with code to process the HomeBUTTON event
041: goto evloop
042: :FindBUTTON
043: info Replace this line with code to process the HomeBUTTON event
044: goto evloop
045: :DelBUTTON
046: info Replace this line with code to process the HomeBUTTON event
047: goto evloop
048: :QuitBUTTON
049: info Replace this line with code to process the HomeBUTTON event
050: goto evloop
051: :close
052: info Replace this line with code to be executed before the dialog closes
053: exit
```

1,Create,Sorted) และอ่านข้อมูลจากแฟ้มเก็บข้อมูลของโปรแกรม (บรรทัดที่ 029) ซึ่งถ้ามีรายการอยู่แฟ้มดังกล่าวก็จะนำมาแสดงบนจอ (บรรทัดที่ 030) แต่ถ้าไม่มีรายการใด ๆ อยู่โปรแกรมก็จะข้ามไป ในชุดคำสั่งช่วงนี้มีการกำหนดตัวแปร %N ขึ้นมาเพื่อใช้ตรวจสอบการบันทึกข้อมูล ซึ่งรายละเอียดจะอยู่ในช่วง :updaterec สำหรับชุดคำสั่งในช่วงนี้จะเกิดการทำงานเพียงครั้งเดียวตอนที่โปรแกรมถูกเรียกขึ้นมาทำงานเท่านั้น สำหรับคำสั่ง List นี้ถือว่าเป็นคำสั่งจัดการพื้นที่ทำงานในหน่วยความจำสารพัดประโยชน์ของ Visual DialogScript เพราะพื้นที่ทำงานที่สร้างด้วยคำสั่งนี้ จะสามารถใช้กับข้อมูลได้ทุกรูปแบบ

- ชุดคำสั่งในช่วง :evloop (บรรทัดที่ 035 ถึง 037) เป็นชุดคำสั่งที่คอยตรวจสอบพฤติกรรมการทำงานของผู้ใช้โปรแกรม โดยคำสั่งทั้งหมดในช่วงนี้ Visual DialogScript จะสร้างขึ้นมาให้หลังออกแบบหน้าจอเสร็จ

- ชุดคำสั่งในช่วง :HomeBUTTON (บรรทัดที่ 043 ถึง 053) เป็นชุดคำสั่งที่จะทำงานเมื่อผู้ใช้คลิกที่ปุ่มคำสั่ง "|<=" ของโปรแกรม โดยจะทำหน้าที่เลื่อนรายการไปยังรายการแรกของข้อมูลที่เก็บอยู่ List โดยใช้คำสั่ง List 1,seek,0 (บรรทัดที่ 050)

- ชุดคำสั่งในช่วง :PrevBUTTON (บรรทัดที่ 055 ถึง 058)

เป็นชุดคำสั่งที่จะทำงานเมื่อผู้ใช้คลิกที่ปุ่มคำสั่ง "<" ของโปรแกรมจะทำหน้าที่เลื่อนรายการไปยังรายการก่อนหน้ารายการที่ปรากฏอยู่บนจอ 1 รายการ

คำสั่ง %S = @index() (บรรทัดที่ 060) เป็นการอ่านข้อมูลรายการปัจจุบันจาก List มาเก็บในตัวแปร %S ส่วนคำสั่ง List ,seek,@Pred(%S) (บรรทัดที่ 062) จะค้นหารายการที่อยู่ก่อนหน้า %S โดยทำงานผ่านฟังก์ชัน @Pred()

- ชุดคำสั่งในช่วง :NextBUTTON (บรรทัดที่ 067 ถึง 077) เป็นชุดคำสั่งที่จะทำงานเมื่อผู้ใช้คลิกที่ปุ่มคำสั่ง ">" ของโปรแกรม ซึ่งจะทำหน้าที่เลื่อนรายการข้อมูลไปยังรายการถัดไป คำสั่งที่ใช้เลื่อนรายการในช่วง :NextBUTTON นี้จะต่างกับการทำงานในช่วง :PrevBUTTON ตรงที่โปรแกรมจะอ่านข้อมูลชุดถัดไปมาเก็บไว้ในตัวแปร %S (บรรทัดที่ 072) ก่อน แล้วจึงค่อยเลื่อนรายการ

- ชุดคำสั่งในช่วง :EndBUTTON (บรรทัดที่ 079 ถึง 087) เป็นชุดคำสั่งที่จะทำงานเมื่อผู้ใช้คลิกที่ปุ่มคำสั่ง "=>|" ของโปรแกรม ซึ่งจะทำหน้าที่เลื่อนรายการไปยังรายการสุดท้ายของข้อมูลที่เก็บใน List โดยจะใช้วิธีนับจำนวนรายการข้อมูลทั้งหมดก่อน แล้วใช้ค่าที่ได้เป็นตัวกำหนดตำแหน่งที่จะเลื่อนไป ซึ่งตำแหน่งของข้อมูลชุดสุดท้ายจะอยู่ที่จำนวนรายการทั้งหมด - 1 เพราะตำแหน่งรายการแรกจะเริ่มที่ 0

- ชุดคำสั่งในช่วง :AddBUTTON (บรรทัดที่ 089 ถึง 098) เป็นชุดคำสั่งที่จะทำงานเมื่อผู้ใช้คลิกที่ปุ่มคำสั่ง "เพิ่ม" ของโปรแกรม จะทำหน้าที่เคลียร์ช่องรับข้อมูลที่ปรากฏบนจอให้ว่างด้วยคำสั่ง Parse (บรรทัดที่ 094) พร้อมกับกำหนดค่าให้ตัวแปร %N มีค่าเป็น 1

- ชุดคำสั่งในช่วง :PostBUTTON (บรรทัดที่ 100 ถึง 108) เป็นชุดคำสั่งที่จะทำงานเมื่อผู้ใช้คลิกที่ปุ่มคำสั่ง "บันทึก" ของโปรแกรม โดยจะทำหน้าที่เสมือนสั่งให้โปรแกรมบันทึกข้อมูลจริง แต่เวลาที่โปรแกรมทำงาน จะมีการบันทึกข้อมูลตลอดเวลาที่มีการเลื่อนหรือเปลี่ยนแปลงข้อมูลบนจอ ซึ่งปุ่มคำสั่งชุดนี้ไม่จำเป็นต้องมีก็ได้

- ชุดคำสั่งในช่วง :FindBUTTON (บรรทัดที่ 110 ถึง 122) เป็นชุดคำสั่งที่จะทำงานเมื่อผู้ใช้คลิกที่ปุ่มคำสั่ง "ค้นหา" ของโปรแกรม โดยจะใช้ฟังก์ชัน @Input() (บรรทัดที่ 115) ในการรับค่าที่ต้องการนำไปค้นในรายการข้อมูลจากผู้ใช้ วิธีการค้นเราจะเลื่อนรายการไปที่รายการแรกก่อน (บรรทัดที่ 116) จากนั้นใช้ฟังก์ชัน @match() ในการค้นหาที่ต้องการ ถ้าพบก็จะนำรายการนั้นมาแสดงบนจอ ถ้าไม่พบโปรแกรมจะแสดงกรอบข้อความดังตัวอย่างในรูปที่ 5

- ชุดคำสั่งในช่วง :DelBUTTON (บรรทัดที่ 124 ถึง 132) เป็นคำสั่งที่จะเกิดการทำงานเมื่อผู้ใช้คลิกที่ปุ่มคำสั่ง "ลบ" ของโปรแกรม จะทำหน้าที่ลบรายการข้อมูลที่ปรากฏบนจอออกจาก List แต่ก่อนลบจะมีการถามความแน่ใจด้วยฟังก์ชัน @ask() ก่อน โดยเราจะดึงข้อความที่อยู่ในช่องรับข้อมูลมาใส่ในกรอบข้อความที่ใช้ถาม ด้วยคำสั่ง @dlgtext() (บรรทัดที่ 128)

รูปที่ 5 กรอบข้อความแสดงข่าวสารกรณีค้นไม่พบ

- ชุดคำสั่งในช่วง :QuitBUTTON และ :close (บรรทัดที่ 134 ถึง 143) สำหรับการทำงานที่ปุ่มคำสั่ง "เลิกงาน" นั้น จะทำหน้าที่เหมือนกับคำสั่งในช่วง :close ที่ Visual DialogScript สร้างไว้ให้ ดังนั้นเราจึงใช้ชุดคำสั่งร่วมกันได้ โดยก่อนจะออกจากโปรแกรม จะทำการจัดเก็บข้อมูลใน List ลงแฟ้มข้อมูลก่อนด้วยคำสั่ง List 1,savefile, addrbook.txt

หมายเหตุ สำหรับชื่อของ Label ที่เกี่ยวข้องกับปุ่มคำสั่งในข้างต้น เป็น Label ที่ Visual DialogScript กำหนดไว้ให้ เราไม่ควรจะเปลี่ยนแปลงใด ๆ เนื่องจากจะทำให้การควบคุมกลไกภายในของ Visual DialogScript เสียไปได้

- ชุดคำสั่งในช่วง :updaterec (บรรทัดที่ 145 ถึง 159) สำหรับงานในช่วงนี้ จะทำหน้าที่เสมือนกับโปรแกรมย่อย สำหรับจัดเก็บข้อมูลที่อยู่บนจอลงใน List โดยโปรแกรมจะตรวจสอบว่ามีข้อมูลอยู่ในช่องต่าง ๆ บนหน้าจอหรือไม่ จากนั้นจะนำข้อความที่อยู่ในแต่ละช่องมาเรียงต่อกันเก็บไว้ในตัวแปร %T (บรรทัดที่ 150) แล้วเปรียบเทียบข้อมูลที่อ่านขึ้นมาครั้งแรก (เก็บใน %Y จากช่วง :showrec) ถ้าตรงกันจะไม่เก็บค่าใด ๆ แต่ถ้าไม่ตรงกัน จะตรวจสอบค่าใน %N ถ้าเป็น 0 จะหมายถึงมีการแก้ไขรายละเอียดในรายการนั้น ๆ เวลาบันทึก เราจะลบข้อมูลเก่าออกก่อนแล้วจึงบันทึก แต่ถ้าเป็น 1 โปรแกรมจะบันทึกข้อมูลทันที

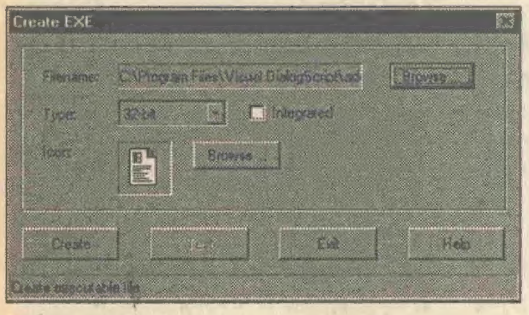
- ชุดคำสั่งในช่วง :showrec (บรรทัดที่ 161 ถึง 169) ชุดคำสั่งในกลุ่มนี้จะทำหน้าที่นำข้อมูลรายการปัจจุบันมาใส่ในช่องรับข้อมูลที่อยู่บนจอ โดยเราจะอ่านข้อมูลจาก List มาเก็บในตัวแปร %Y แล้วใช้คำสั่ง parse นำค่าที่เก็บใน %Y ไปใส่ในช่องรับข้อมูลแต่ละช่องตามลำดับ สำหรับ :showrec นี้จะเป็นลาเบลแบบโปรแกรมย่อยเหมือนกับ :updaterec

การสร้าง Executable File

สำหรับโปรแกรมที่สร้างด้วย Visual DialogScript นอกจากเราจะสามารถเรียกโปรแกรมขึ้นมาทำงานด้วยปุ่มคำสั่ง ▶ ของ Visual DialogScript แล้ว เรายังสามารถคอมไพล์และลิงค์เป็น Executable File เพื่อไปใช้งานอิสระได้ด้วย โดยเราเพียงคลิกที่ปุ่มคำสั่ง ⚙ ที่อยู่บนแถบเครื่องมือของโปรแกรม Visual DialogScript จะเรียกไดอะล็อก Create EXE ขึ้นมาทำงาน เพื่อให้เราจัดองค์ประกอบของโปรแกรม

Visual DialogScript 2.2

เครื่องมือสร้างโปรแกรมอย่างง่าย ๆ



รูปที่ 6 ไดอะล็อก Create EXE สำหรับสร้าง Executable File

ดังตัวอย่างในรูปที่ 6 และเมื่อเราคลิกที่ปุ่มคำสั่ง OK ของไดอะล็อกดังกล่าว Visual DialogScript จะสร้าง Executable file ให้ทันที

ถ้าดูวิธีการสร้างชุดคำสั่งของ Visual DialogScript ที่ผมนำมาเสนอนี้ จะรู้สึกว่าไม่ใช่เรื่องยากเย็นอะไร คนที่เขียนแบตช์ไฟล์อยู่ประจำคงจะพอเห็นแนวทางที่จะนำเครื่องมือชุดนี้ไปใช้งาน ส่วนท่านที่กำลังฝึกหัดเขียนโปรแกรม จะลองใช้เครื่องมือชุดนี้สร้างโปรแกรมเล็ก ๆ ไว้ใช้งาน ก็ไม่เสียหายอะไรครับ

ลิสติงที่ 2 ADDRESS.DSC

```
001: title Test
002: REM สร้างส่วนหน้าจอของโปรแกรม
003: DIALOG CREATE,โปรแกรมเก็บเบอร์โทรบุคคล,-
     1,0,344,185,-
004: TEXT(TEXT1;8;10;;;ชื่อ :),-
005: EDIT(NAME;8;56;192),-
006: TEXT(TEXT2;32;10;;;นามสกุล :),-
007: EDIT(LASTNAME;32;56;192),-
008: TEXT(TEXT3;56;10;;;โทร :),-
009: EDIT(TEL;56;56;192),-
010: TEXT(TEXT4;80;10;;;หมายเหตุ),-
011: EDIT(NOTE;80;56;192),-
012: BUTTON(Home;120;14;38;;|<=),-
013: BUTTON(Prev;120;54;38;;<=),-
014: BUTTON(Next;120;94;38;;=>),-
015: BUTTON(End;120;134;38;;=>|),-
016: BUTTON(Add;8;264;;;เพิ่ม),-
017: BUTTON(Post;33;264;;;บันทึก),-
018: BUTTON(Find;58;264;;;ค้นหา),-
019: BUTTON(Del;83;264;;;ลบ),-
020: BUTTON(Quit;120;264;;;เลิกงาน),-
021: Status(SP;No data)
022:
023: REM สร้างพื้นที่ทำงานแบบเรียงข้อมูลอัตโนมัติ
024: REM แล้วอ่านข้อมูลจากแฟ้มข้อมูลมาใส่พื้นที่ที่เตรียมไว้
025: REM พร้อมกับเรียกขึ้นมาแสดงบนจอ (showrec)
026: REM ----------
027: list 1,create,sorted
028: %N = 1
029: list 1,loadfile,addrbook.txt
030: if @not(@zero(@count(1)))
031: gosub showrec
032: %N = 0
033: end
034:
035: :evloop
036: wait event
037: goto @event()
038: REM ----------
039: REM evloop ทำหน้าที่รอรับการสั่งงานของผู้ใช้
040: REM เป็นกลไกที่สร้างโดย Visual
041: REM DialogScript
042:
043: :HomeBUTTON
044: REM HomeBUTTON ทำหน้าที่เลื่อนตัวชี้รายการไปยัง
045: REM รายการแรกของข้อมูลทั้งหมด
046: REM ----------
047: gosub updaterec
048: %S = @index(1)
049: if @not(@zero(%S))
050: list 1,seek,0
051: end
052: gosub showrec
053: goto evloop
054:
055: :PrevBUTTON
056: REM PrevBUTTON ทำหน้าที่เลื่อนตัวชี้รายการไปยัง
057: REM รายการก่อนหน้านี้
058: REM ----------
059: gosub updaterec
060: %S = @index(1)
061: if @not(@zero(%S))
062: list 1,seek,@pred(%S)
063: end
064: gosub showrec
065: goto evloop
066:
067: :NextBUTTON
068: REM NextBUTTON ทำหน้าที่เลื่อนตัวชี้รายการไปยัง
069: REM รายการถัดไป
070: REM ----------
071: gosub updaterec
072: %S = @succ(@index(1))
073: if @greater(@count(1),%S)
074: list 1,seek,%S
075: end
076: gosub showrec
077: goto evloop
078:
079: :EndBUTTON
080: REM EndBUTTON จะทำหน้าที่เลื่อนไปยังรายการ
081: REM สุดท้ายของรายการข้อมูลทั้งหมด
082: REM ----------
083: gosub updaterec
084: %S = @count(1)
085: list 1,seek,@pred(%S)
086: gosub showrec
087: goto evloop
088:
089: :AddBUTTON
090: REM AddBUTTON จะทำหน้าที่สร้างพื้นที่เพื่อรับข้อมูล
091: REM รายการใหม่ แต่จะยังไม่จัดเก็บ
092: REM ----------
093: gosub updaterec
094: parse NAME;LASTNAME;TEL;NOTE,||||
095: %N = 1
096: dialog set,SP,New record
097: dialog focus,NAME
098: goto evloop
099:
100: :PostBUTTON
101: REM PostBUTTON จะทำงานเมื่อผู้ใช้คลิกที่ปุ่มคำสั่ง บันทึก
102: REM โดยจะไปทำงานที่ updaterec
103: REM ----------
104: gosub updaterec
105: %N = 0
```

ลิสติงที่ 2 (ต่อ)

```
106: dialog set,SP,Save record
107: dialog focus,NAME
108: goto evloop
109:
110: :FindBUTTON
111: REM FindBUTTON จะทำงานเมื่อผู้ใช้คลิกที่ปุ่มคำสั่ง ค้นหา
112: REM ————————————————
113: %S = @index(1)
114: REM เรียกไดอะล็อกเพื่อรับคำจากผู้ใช้
115: %G = @input(Enter search text:,%G)
116: list 1,seek,0
117: if @not(@match(1,%G))
118: info No match for %G found
119: list 1,seek,%S
120: end
121: gosub showrec
122: goto evloop
123:
124: :DelBUTTON
125: REM DelBUTTON จะทำงานเมื่อผู้ใช้คลิกที่ปุ่มคำสั่ง ลบ
126: REM เพื่อลบรายการที่ปรากฏอยู่บนจอ
127: REM ————————————————
128: if @ask(Delete record for @dlgtext(NAME)?)
129: list 1,delete
130: end
131: gosub showrec
132: goto evloop
133:
134: :QuitBUTTON
135: REM QuitBUTTON จะทำงานเมื่อผู้ใช้คลิกที่ปุ่มคำสั่งเลิกงาน
136: REM โดยคำสั่งจะใช้ร่วมเดียวกับ :close
137: REM ————————————————
138: :close
139: REM close จะทำหน้าที่ปิดการทำงานของโปรแกรม
140: REM ————————————————
141: gosub updaterec
142: list 1,savefile,addrbook.txt
143: exit
144:
145: :updaterec
146: REM updaterec จะทำหน้าที่บันทึกข้อมูลที่อยู่บน
147: REM จอภาพลงใน List
148: REM ————————————————
149: if @dlgtext(NAME)
150: %T = @dlg(NAME)|@dlg(LASTNAME)|@dlg(TEL)|@dlg(NOTE)|
151: if @not(@equal(%Y,%T))
152: if @zero(%N)
153: list 1,delete,%S
154: end
155: list 1,add,%T
156: end
157: end
158: %N = 0
159: exit
160:
161: :showrec
162: REM showrec จะเป็นส่วนที่ทำหน้าที่นำข้อมูลจากแฟ้มข้อมูล
163: REM มาแสดงไว้ในช่องต่าง ๆ ของโปรแกรม
164: REM ————————————————
165: %Y = @item(1)
166: parse NAME;LASTNAME;TEL;NOTE,%Y
167: dialog set,SP,Record @succ(@index(1)) of @count(1)
168: dialog focus,NAME
169: exit
```